Tempered Particle Filter Performance

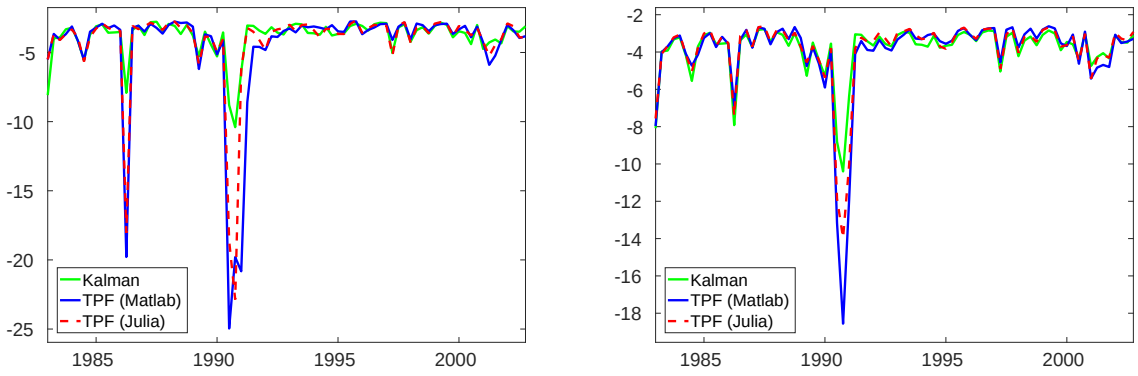
Ethan Matlin and Reca Sarfati

July 26, 2017

1 An-Schorfheide

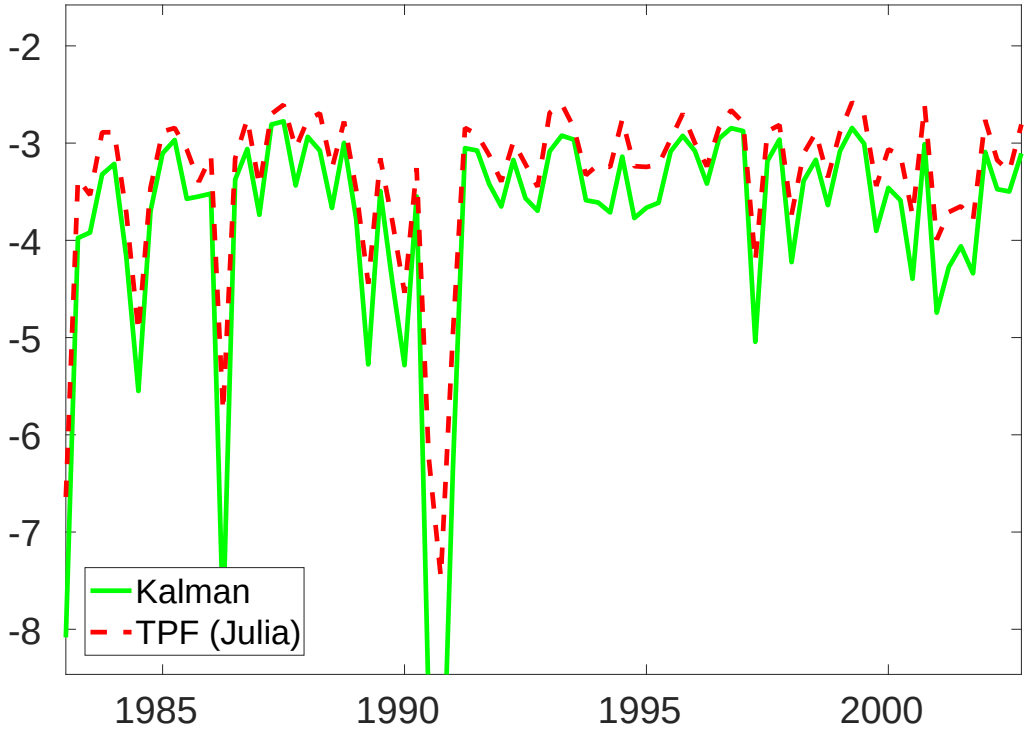
Figure 1: Comparison of Kalman, TPF (Matlab), and TPF(Julia)

500 Particles 4000 Particles



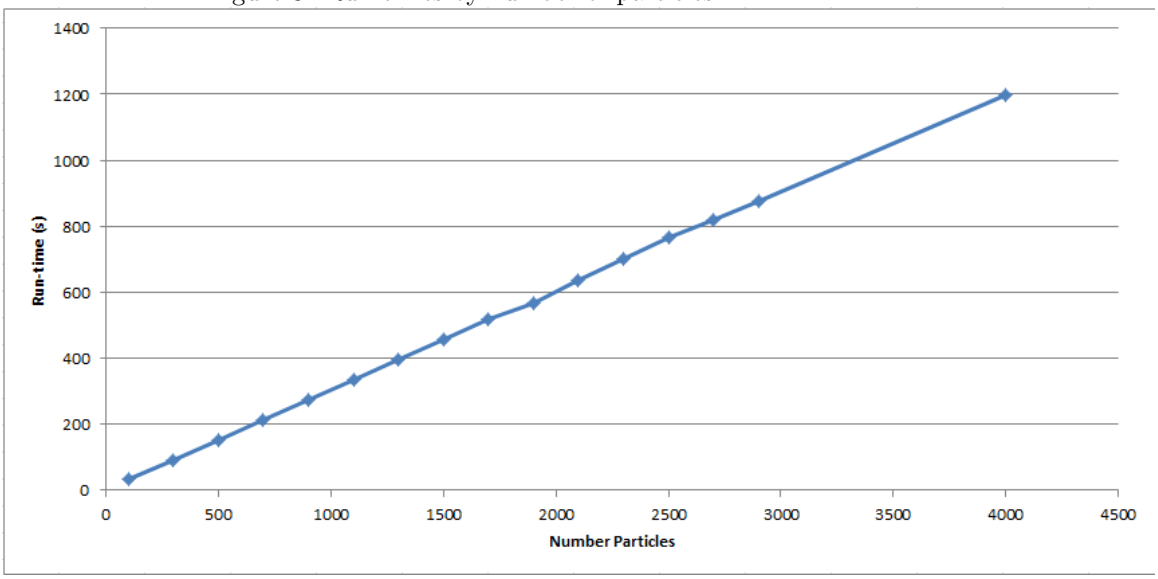
Note: Fixed ϕ schedule, deterministic sampling, random matrices fixed, using us.txt (Shorfheide's data set) and system matrices computed from Matlab code. Difference between TPF (Matlab) and TPF (Julia) arise due to differences in the numerical solver between Julia and Matlab, rounding differences, and eigenvalue computation.

Figure 2: Comparison of Kalman and TPF



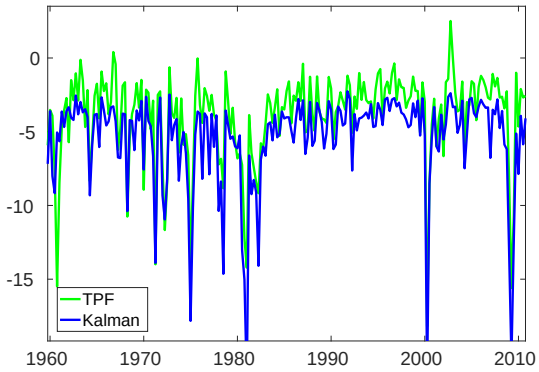
Note: 4000 particles. Regular implementation of the filter (randomness,etc.)

Figure 3: Run-times by number of particles

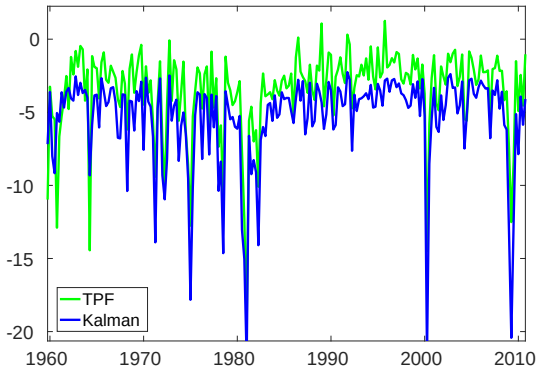


2 Smets-Wouters

4000 Particles, 2 MH

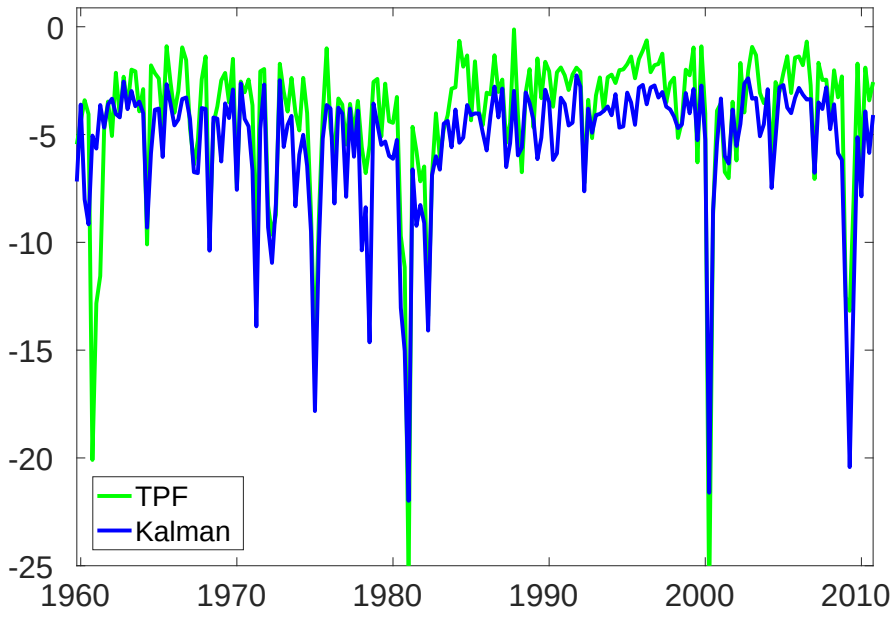


10,000 Particles, 2MH



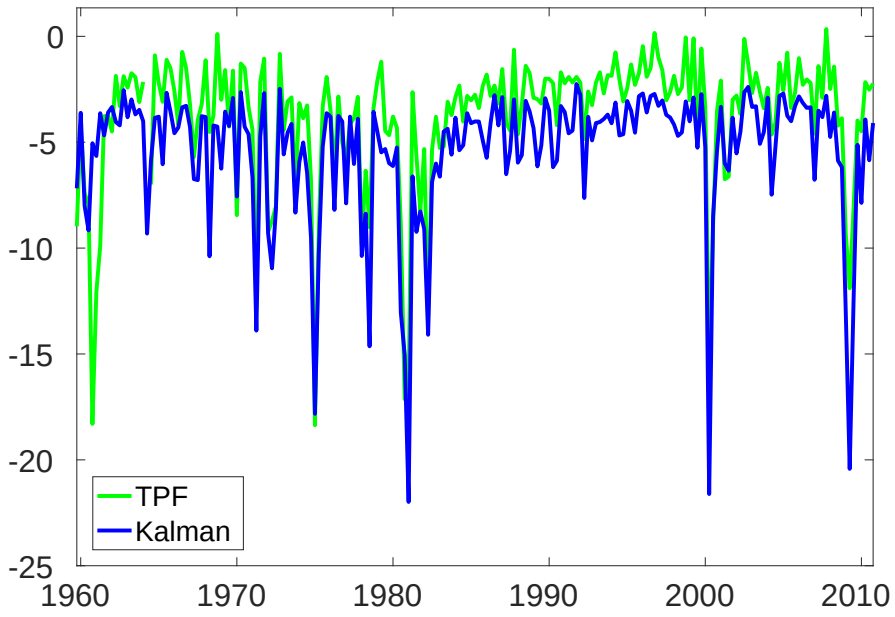
Note: Vertical axist in log-likelihood units. $r^* = 2$

Figure 4: Kalman vs. TPF (10,000 particles, 5 MH steps)



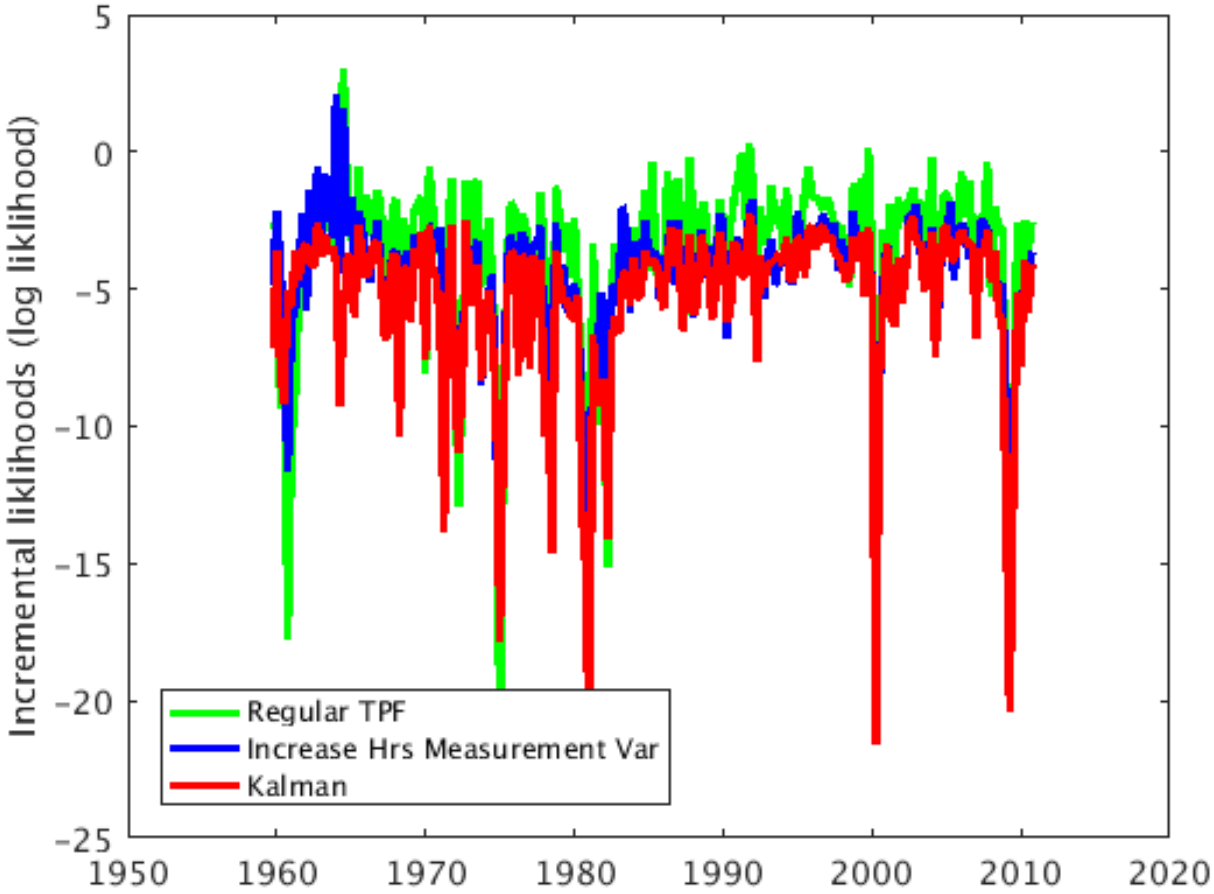
Note: Vertical axis in log-likelihood units. 4,000 particles. 5 MH steps per mutation. $r^* = 2$

Figure 5: Kalman vs. TPF (4,000 particles, 10 MH steps)



Note: Vertical axis in log-likelihood units. 4,000 particles. 10 MH steps per mutation. $r^* = 2$

Figure 6: Comparison of TPF, inflated variance TPF, and Kalman filter



Note: TPF has measurement error variance $S/10$ for all series (where S is the sample standard deviation of the series). Inflated variance TPF has variance $\approx 5.7S$ for hours worked and $S/10$ for all other series. Vertical axis in log-likelihood units. 4,000 particles. 10 MH steps per mutation. $r^* = 2$.

Figure 7: Runtime by number of particles

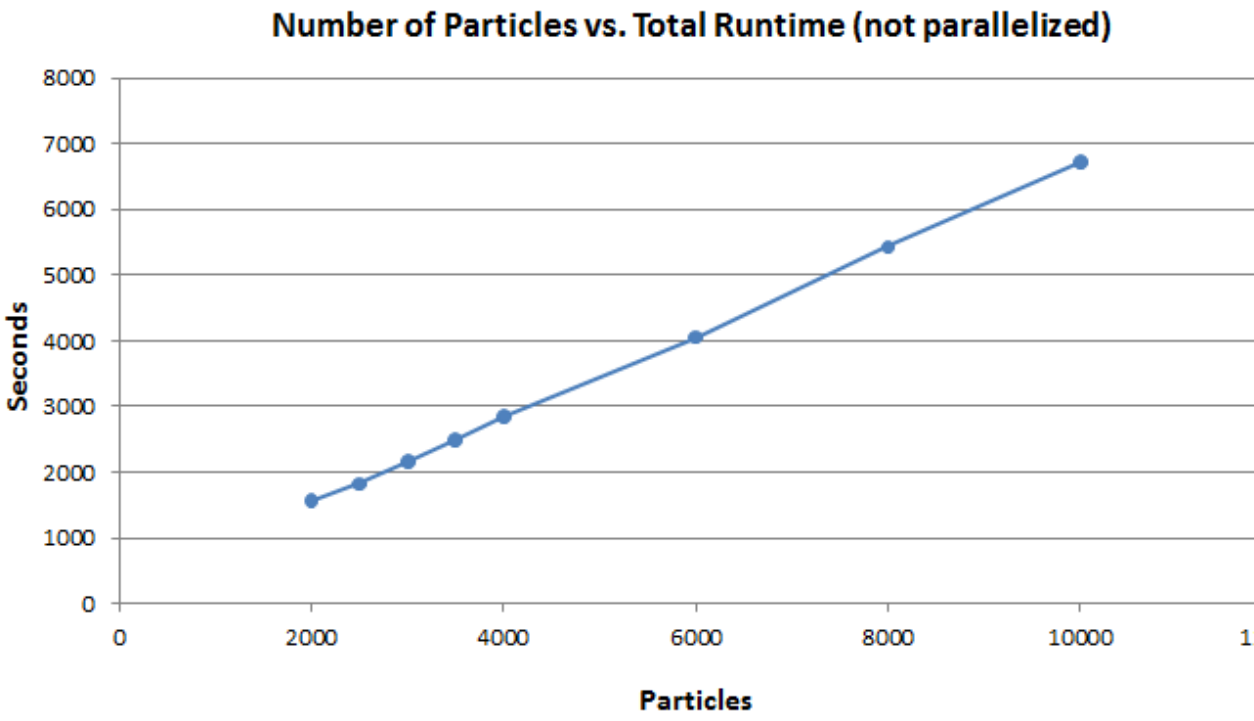
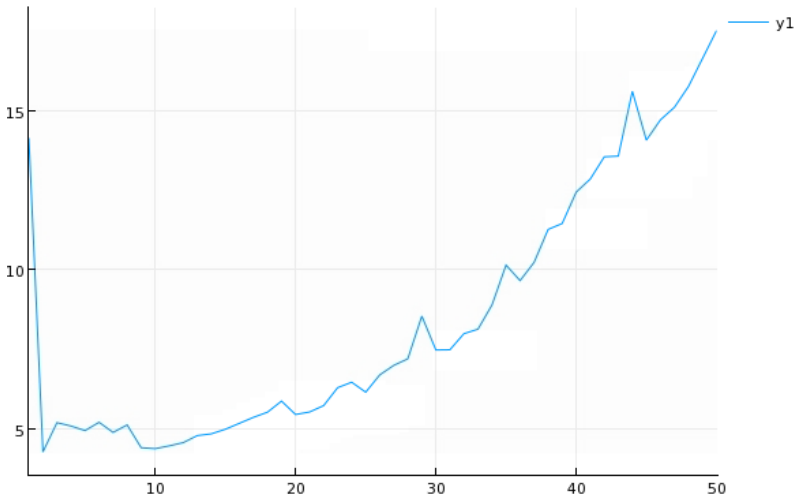


Figure 8: Increase in runtime per time step (10 calls to mutation per time step)



3 Problems with Parallelism

Figure 9: Timing and Workers

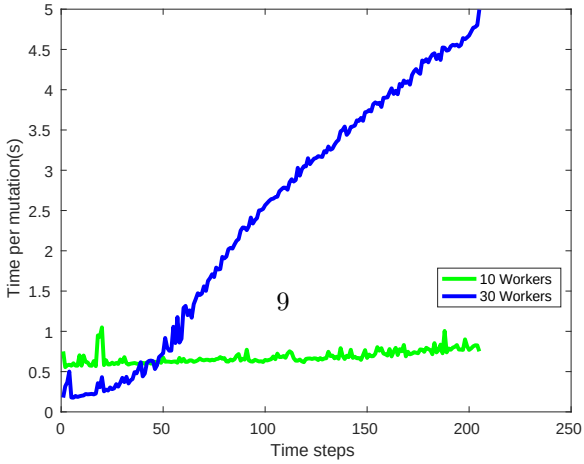
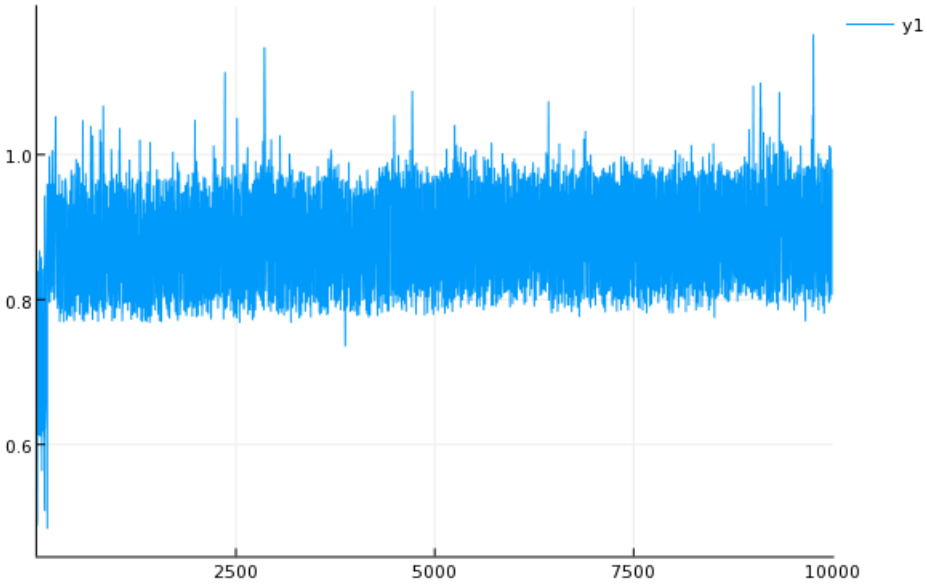
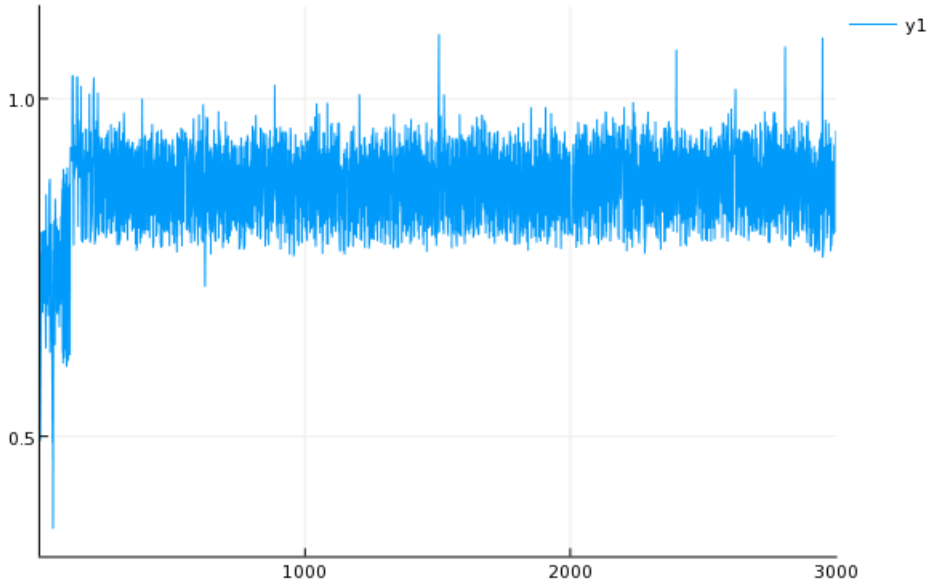


Figure 10: Run-Time



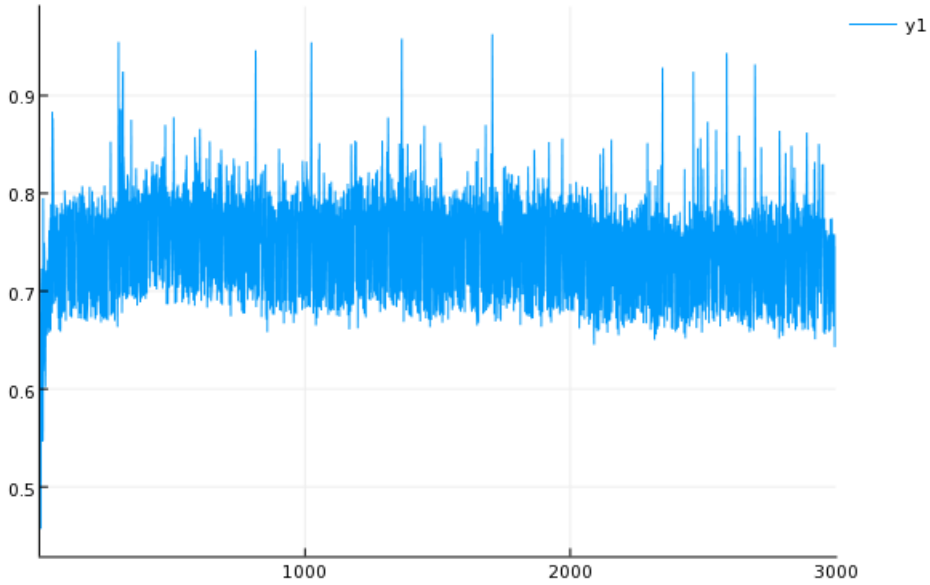
Note: This graph plots run-time of a function (run in parallel using @sync @parallel (hcat)) that takes in a SmetsWouters model object, two 100×100 matrices (A and B), and an integer (x from 1-4000) and returns x^2 and the transpose of $A \times B$. 10 workers. Parallelism is run over all values of x (1:4000). It tests whether passing in the model object causes the increase in runtime.

Figure 11: Run-Time



Note: This graph plots run-time of a function (run in parallel using @sync @parallel (hcat)) that takes in a SmetsWouters model object, two 100×100 matrices (A and B), and an integer (x from 1-4000) which comes from indexing into an array and returns x^2 and the transpose of $A \times B$. 10 workers. Parallelism is run over all values of x (1:4000). It tests whether syncing based on array index is causing the problem.

Figure 12: 30 workers with the simple test-case



Note: This graph plots run-time of a function (run in parallel using @sync @parallel (hcat)) that takes in a SmetsWouters model object, two 100×100 matrices (A and B), and an integer (x from 1-4000) which comes from indexing into an array and returns x^2 and the transpose of $A \times B$. 30 workers. Parallelism is run over all values of x (1:4000). It tests whether syncing based on array index is causing the problem.